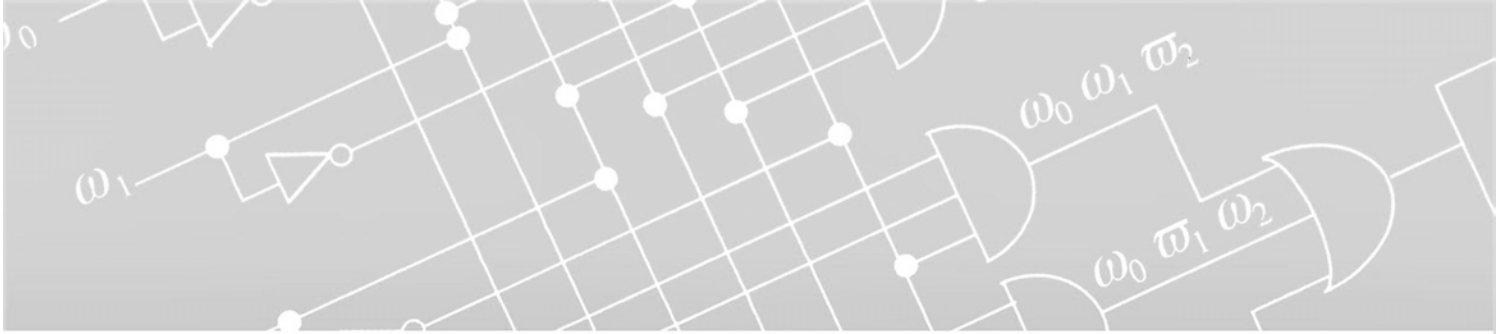
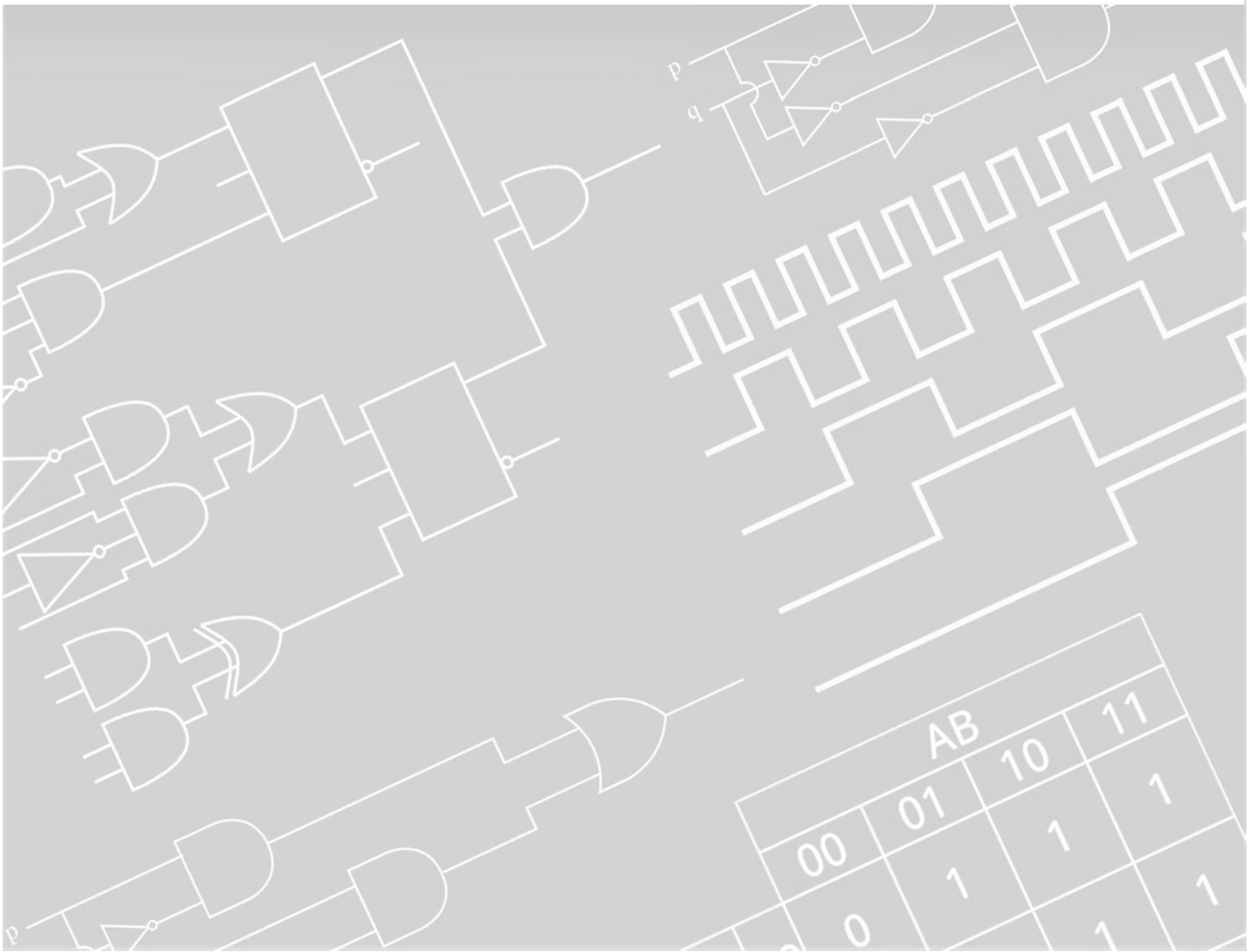


Shiraz University
School of Electrical and Computer Engineering

Logic Circuits Laboratory



Experiment 1

Basic Logic Gates

Objective: Introduction to digital ICs and measuring simple gate characteristics

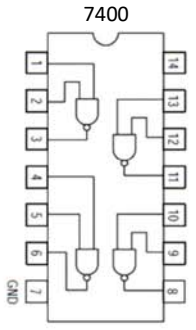
***Required Equipment*:** KL-31001 (Main Unit), Breadboard, Basic Gates (7400, 7404, …), Oscilloscope.

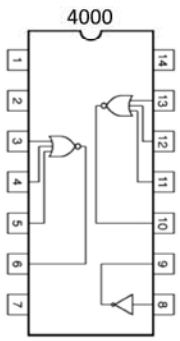
BACKGROUND INFORMATION

A variety of digital integrated circuits (ICs) are used in the experiments of this manual. For a detailed description of the performance of each IC, you can refer to data-sheets available on the web or in data books published by the chip manufacturer. The sheets specify the function of each pin of the IC package and provide detailed data on chip performance.

In the following, we review a few basic points about digital ICs. **You are expected to thoroughly study this material before the lab**.

Digital ICs





The most commonly used logic families are the TTL (transistor–transistor logic) and CMOS (complementary metal-oxide-semiconductor) families. As every other chip, digital ICs have a part number printed on them. The part number of TTL chips begins with 74 and they are called 7400 series (or 74xx series). For example, the first part number in the series, the 7400, designates a device containing four two-input NAND gates, or the 7404 is a hex inverter (contains six inverters).

Originally, the part number of CMOS digital ICs began from 4000, and so were called 4000 (or 4xxx) series. The first part number in the series, the IC type 4000, has two 3-input NOR gates and one inverter.

However, later they adopted the 7400 numbering and the CMOS subfamilies 74C, 74HC, 74AC, … appeared. These series are pin and function compatible with TTL series. For example, CMOS IC type 74C04 has six inverters with the same configuration as TTL type 7404.

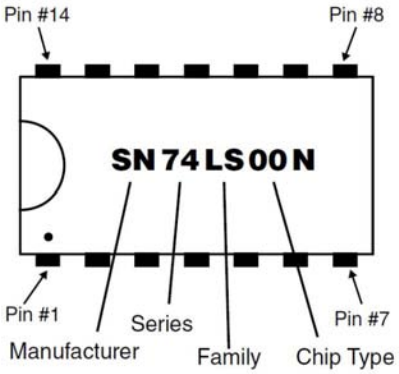
Each of the TTL and CMOS logic families has several subfamilies which differ in voltage range, propagation delay, power consumption, and so on. The following table summarizes some of the important logic families.

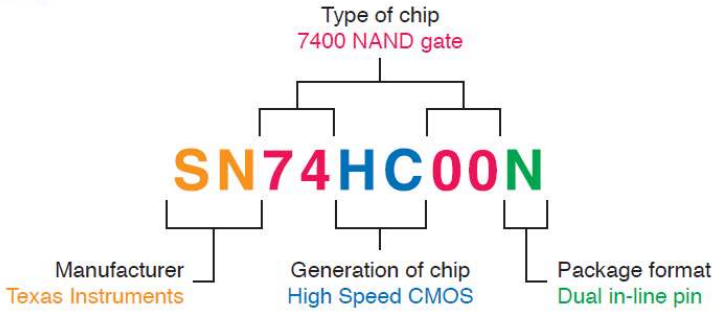
Comparison of various performance characteristics of important logic families.

Logic family		Supply voltage (V)	Typical propagation delay (ns)	Worst-case noise margin (V)	Speed–power product (pJ)	Maximum flip-flop toggle frequency (MHz)
TTL	Standard	4.5 to 5.5	17	0.4	100	35
	L	4.5 to 5.5	60	0.3	33	3
	H	4.5 to 5.5	10	0.4	132	50
	S	4.5 to 5.5	5	0.3	57	125
	LS	4.5 to 5.5	15	0.3	18	45
	ALS	4.5 to 5.5	10	0.3	4.8	70
	AS	4.5 to 5.5	4.5	0.3	13.6	200
	F	4.5 to 5.5	6	0.3	10	125
CMOS	4000	3 to 15	150	1.0	5	12
	74C	3 to 13	50	1.4	5	12
	74HC	2 to 6	8	0.9	1.4	40
	74HCT	4.5 to 5.5	8	1.4	1.4	40
	74AC	2 to 6	4.7	0.7	0.37	100
	74ACT	4.5 to 5.5	4.7	0.72.9	0.37	100

In brief, TTL chips have a narrow power supply range (about 5 V) and high power consumption (~mW). On the other hand, the wider voltage range, wider noise margin and very low power dissipation (~µW) of CMOS chips are accompanied by lower speed (higher propagation delay). However, newer generations of CMOS chips (e.g., high-speed CMOS 74HC series) have overcome this disadvantage.

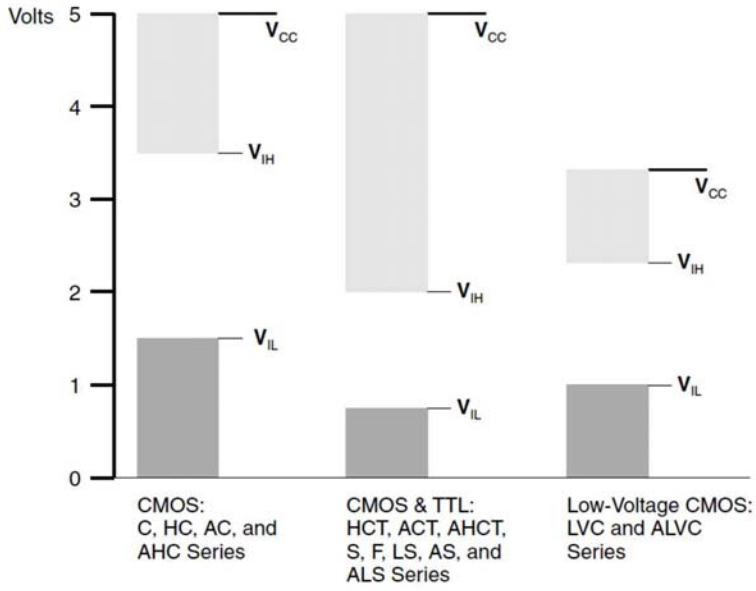
As noted above, aside from the IC type (e.g., NAND, NOR, XOR, …) the logic family of the IC can also be identified from the part number printed on it (see figures below).





In the following, a brief description of some the most important characteristics of logic families is provided.

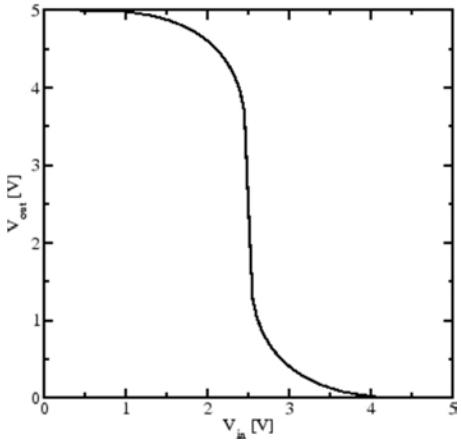
a) ***Logic Levels and Noise Margin***: Digital chips employ two voltages to represent two possible states. These voltages are called logic levels and can be used to represent the two states of Boolean algebra as well as the two digits of binary arithmetic.
In TTL logic, a voltage exceeding +2 V is called *high*, while a voltage less than +0.8 V is called *low*. To ensure *noise margin*, TTL outputs are guaranteed to put out at least +2.5 V in the high state and at most +0.4 V in the low state.



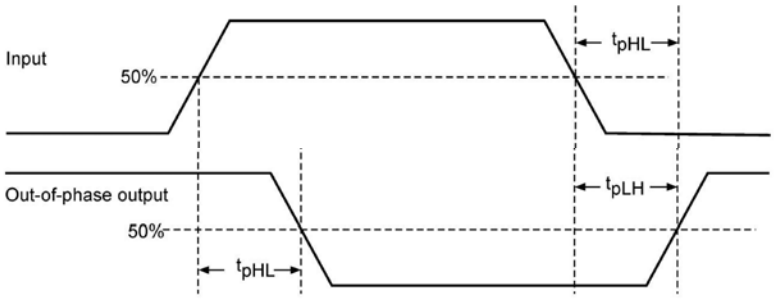
Logic levels for various 7400-family lines. V_{CC} is the most positive voltage; V_{IL} and V_{IH} are the maximum input low and minimum input high voltages.

While TTL chips are always powered from a +5 V supply, many CMOS chips are tolerant of supply voltages ranging from +2 to +6 V (in HC and AC series) or even a wider range in older series. It is therefore convenient to reference CMOS logic levels to the power-supply voltage V_{CC}. The minimum input voltage interpreted as a CMOS high (V_{IH}) equals $0.7 \times V_{CC}$, while the maximum input voltage interpreted as a CMOS low (V_{IL}) equals $0.3 \times V_{CC}$ (In the above figure it is assumed that V_{CC}=5 V for CMOS). The output voltages V_{OH} and V_{OL} will vary with supply voltage as well.

b) ***Transfer Characteristic***: The voltage transfer characteristic of a logic gate is simply a plot of the gate output voltage V_{OUT} versus the gate (slowly varying) input voltage V_{IN}. The following figure shows the transfer characteristic for a typical CMOS inverter. As you can see the transition between the HIGH and LOW levels is not very sharp.



c) ***Propagation Delay***: t_{pLH} is the time delay between the input and output waveforms with the output changing from LOW to HIGH; t_{pHL} is the time delay between the input and output waveforms with the output changing from HIGH to LOW (see figure below).



The average propagation delay time is calculated as the average of these two delays.

PRELAB

1) Consider the ring configuration of an odd number (*n*) of inverters. The following figure shows one such circuit using three inverters.

i. What is the waveform at the output?
ii. Derive a relation between the frequency of the output signal, the average propagation delay of gates, and *n*.

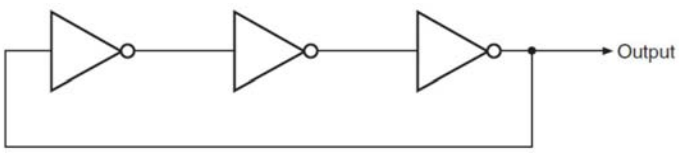


Figure 1

2) Show how a half-adder can be built using only NAND gates.

LAB

Introduction

All experiments in the Logic Circuit Lab use KL-31001 digital logic set. It contains a complete set of digital inputs, output displays, clocks, and the required power supplies. Experiments are performed by putting the breadboard or the specified module on this unit.

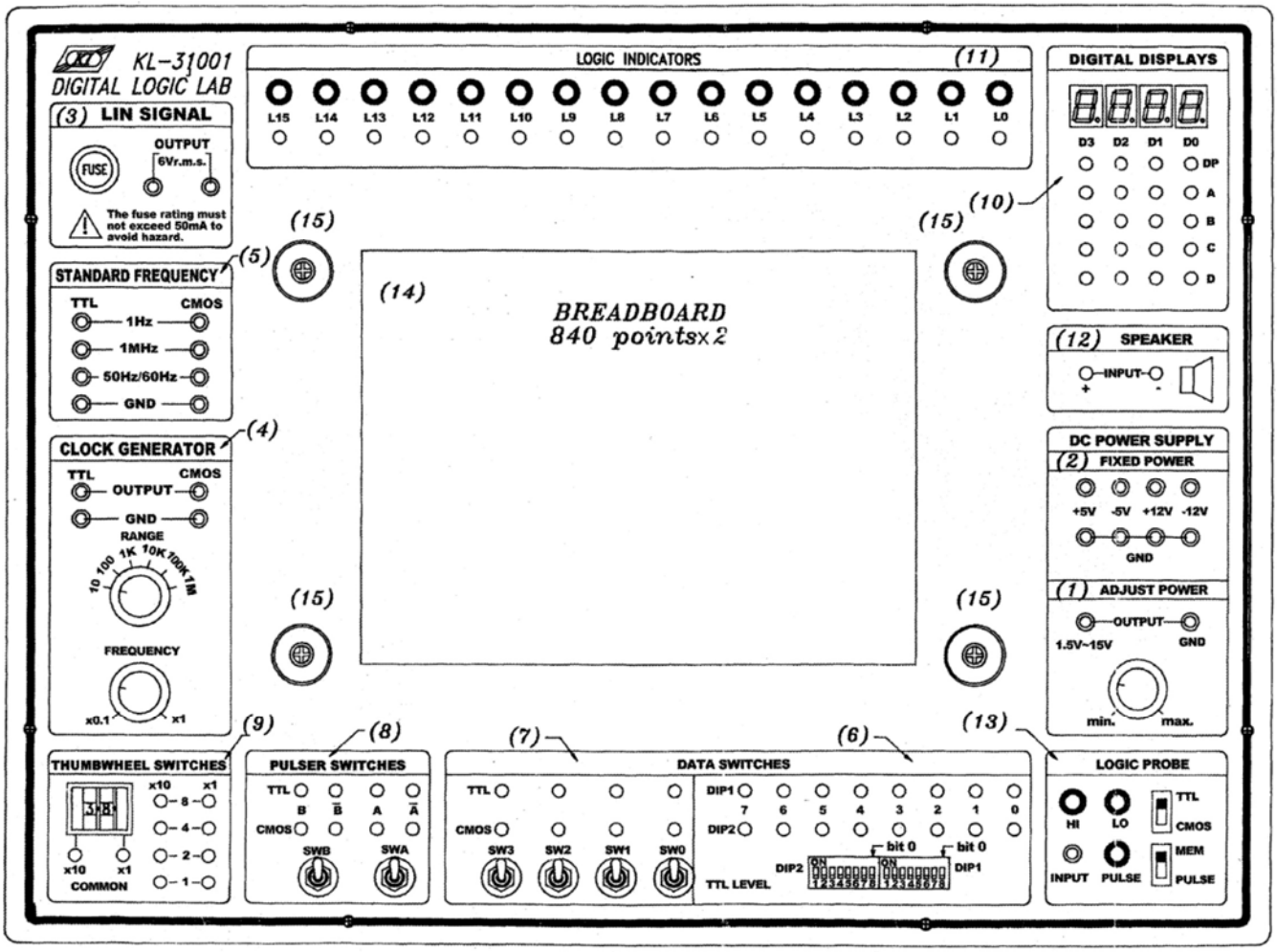
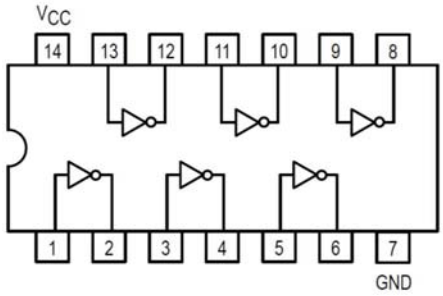


Figure 2

Procedure

> ***Note***: – *When you finish each part of the Lab procedure, before proceeding to the next part, make sure that your TA or instructor checks your results and you get the mark for that part.*
> – *Record the results and waveforms you obtain in each part in your lab report.*

1) Using two 74LS04 ICs, cascade 9 inverters on the breadboard and form a ring oscillator as shown in Figure 1 (page 4). The pinout of 7404 is shown below.



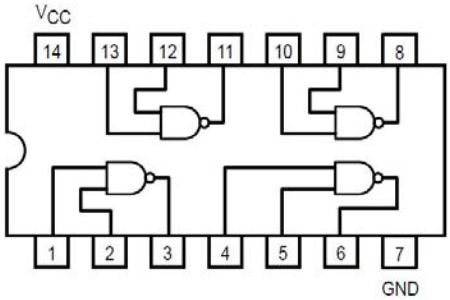
Connect the +5 V and GND of the power supply (on KL-31001) to V_{CC} and GND of the ICs. Apply the output of the ring configuration to one of the oscilloscope channels and observe the waveform.

Attention: The ground of oscilloscope and power supply should be connected together.

Plot this waveform in your lab report and from it determine the average propagation delay of the TTL inverter.
If it is difficult to observe the waveform or measure the delay, cascade 11 inverters.

2) Repeat the previous part for CMOS inverters. Use one 74C04 IC and cascade 3 or 5 inverters.

3) Place a 74LS00 NAND IC on the breadboard and connect it to the power supply.



Consider one of the four NAND gates with inputs *x* and *y* and output *z*. By leaving inputs open (which is also called float) or connected to HIGH and LOW voltages, complete the following table:

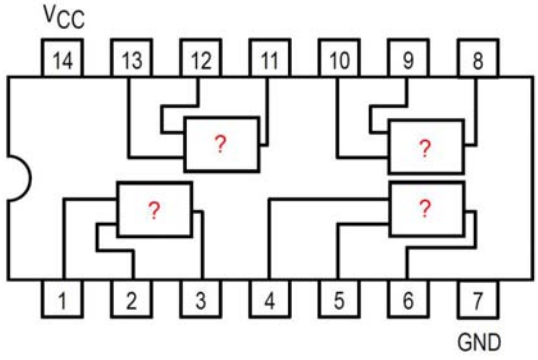
x	*y*	*z*
1	open	
open	0	
0	open	
open	open	

The floating input of TTL family devices behaves as if which logic (HIGH or LOW) has been applied to the input?

4) Using two 74LS00 ICs, implement the half-adder you designed in the **Prelab** part on the breadboard. To test your half-adder as a two-input/two-output combinational circuit, use switches SW0 and SW1 as inputs and connect the two outputs to logic indicators (LEDs) L0 and L1 on KL-31001.

5) You are given another IC (except 7400 and 7404). Write its part number here: _____

This IC contains four identical two-input gates and has the following schematic. By applying appropriate inputs and constructing the truth table, determine the logical function of this IC.



QUESTIONS

***Note**: Answer the following questions at the end of your lab report.*

1) From the datasheet of the inverter ICs (both TTL and CMOS), find the value of the propagation delay and compare it with the result you obtain in parts (1) and (2). Print the page of datasheet that contains the delay values and attach it to your report.

2) What is the definition of I_{IL} and I_{IH} for a gate? From datasheet, find I_{IL} and I_{IH} for both TTL and CMOS inverters and also for 74LS00.

3) What is the definition of *fan-in* and *fan-out* for a gate? From datasheet, find their values for both TTL and CMOS inverters.

4) Consider the configuration in Figure 1 with an *even* number of inverters. What will be the output waveform in this case? Can you suggest a method to measure the gate propagation delay using this configuration?

Experiment 2

Combinational Logic

Objective: Combinational circuit design, NAND and NOR implementation, Understanding the construction and operational principles of digital comparators.

***Required Equipment*:** KL-33002 module.

BACKGROUND INFORMATION

Implementation with One Gate Type:

The fewer and less expensive the parts required to build a circuit, the less it will cost. Therefore, an important goal in circuit design is minimizing the hardware required and the associated cost of that hardware.

It is common practice to implement an expression using only one type of gate—a NAND or NOR gate. The NAND and NOR gates are the most "naturally" implemented function at the internal level of the chip.

A method to implement a circuit using only NAND or NOR gates is to use the gate equivalency rules and convert a function implemented with AND and OR gates to a NAND or NOR implementation. The following equivalent gate representations are useful in design with one gate type.

Invert Function with NAND and NOR

NAND Gate Representations

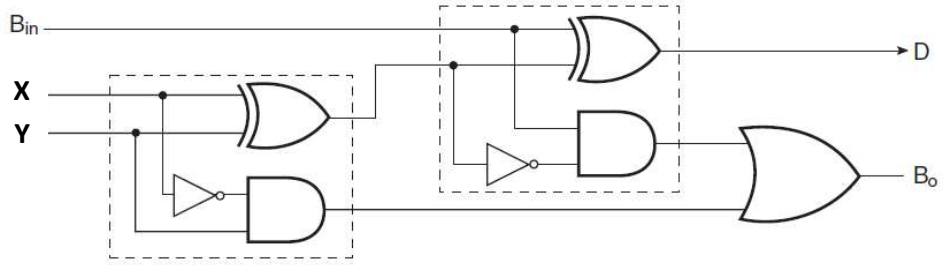
NOR Gate Representations

Full subtractor

The full subtractor is a combinational circuit with three inputs **X**, **Y**, and $\mathbf{B_{in}}$ that computes $\mathbf{X-Y-B_{in}}$ and generates two output bits: the difference **D** and borrow-out $\mathbf{B_{out}}$. ($\mathbf{B_{in}}$ is the borrow from the next less significant digit. That is, it is equal to 1 if a '1' has already been borrowed by the previous adjacent lower bit; otherwise it is 0).

Since we are subtracting **Y** and $\mathbf{B_{in}}$ from **X**, a borrow-out needs to be generated when $\mathbf{X < Y+B_{in}}$. Therefore, the full subtractor generates a borrow-out ($\mathbf{B_{out}}$) when it needs to borrow from the next digit. When a borrow out is generated, 2 is added in the current digit (This is similar to the subtraction algorithm in decimal. Instead of adding 2, we add 10 when we borrow). Therefore, $\mathbf{D=X-Y-B_{in}+2B_{out}}$.

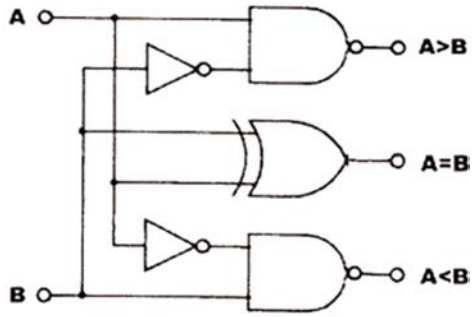
The following figure shows the circuit of the full subtractor (it is easily obtained from the truth table of the full subtractor).



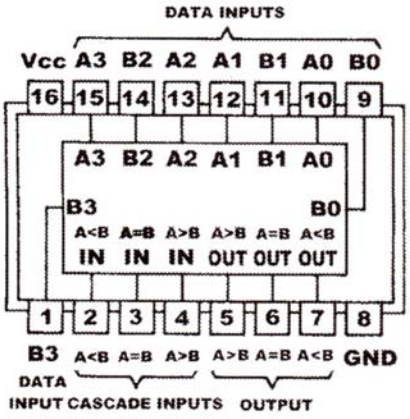
Comparators

At least two numbers are required to perform any comparison. The simplest form of comparator has two inputs. If the two inputs are called A and B there are three possible outputs: A>B; A=B; A<B.

A 1-bit comparator is shown in the following figure.



In practice 4-bit comparators are used most often. The diagram and function table of a typical comparator IC is shown below. As you can see, three inputs are also provided for cascading such 4-bit comparators and building larger ones.



FUNCTION TABLES

COMPARING INPUTS				CASCADING INPUTS			OUTPUTS		
A3,B3	A2,B2	A1,B1	A0,B0	A>B	A<B	A=B	A>B	A<B	A=B
A3>B3	X	X	X	X	X	X	H	L	L
A3<B3	X	X	X	X	X	X	L	H	L
A3=B3	A2>B2	X	X	X	X	X	H	L	L
A3=B3	A2<B2	X	X	X	X	X	L	H	L
A3=B3	A2=B2	A1>B1	X	X	X	X	H	L	L
A3=B3	A2=B2	A1<B1	X	X	X	X	L	H	L
A3=B3	A2=B2	A1=B1	A0>B0	X	X	X	H	L	L
A3=B3	A2=B2	A1=B1	A0<B0	X	X	X	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0	H	L	L	H	L	L
A3=B3	A2=B2	A1=B1	A0=B0	L	H	L	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0	X	X	H	L	L	H
A3=B3	A2=B2	A1=B1	A0=B0	H	H	L	L	L	L
A3=B3	A2=B2	A1=B1	A0=B0	L	L	L	H	H	L

PRELAB

1) Design a circuit that accepts a 4-bit BCD digit (0-9) as input and produces a 2-bit quotient for a division by 3. This circuit will have four inputs –the 4 bits of the BCD digit– and two outputs for the quotient (For example, input=0111 → output=10).

a) Derive the truth table that illustrates the behavior of this circuit.

b) Using K-map (and don't cares!), reduce each of the output equations to its simplest form.

c) Design the circuit for the most significant bit (MSB) of the output using only NOR gates (and inverters) and for the least significant bit (LSB) of the output using only NAND gates.

2) Design the full subtractor circuit using only NAND and NOR gates as follows:

a) Convert the AND and OR gates in the full subtractor circuit to **NOR** gates.

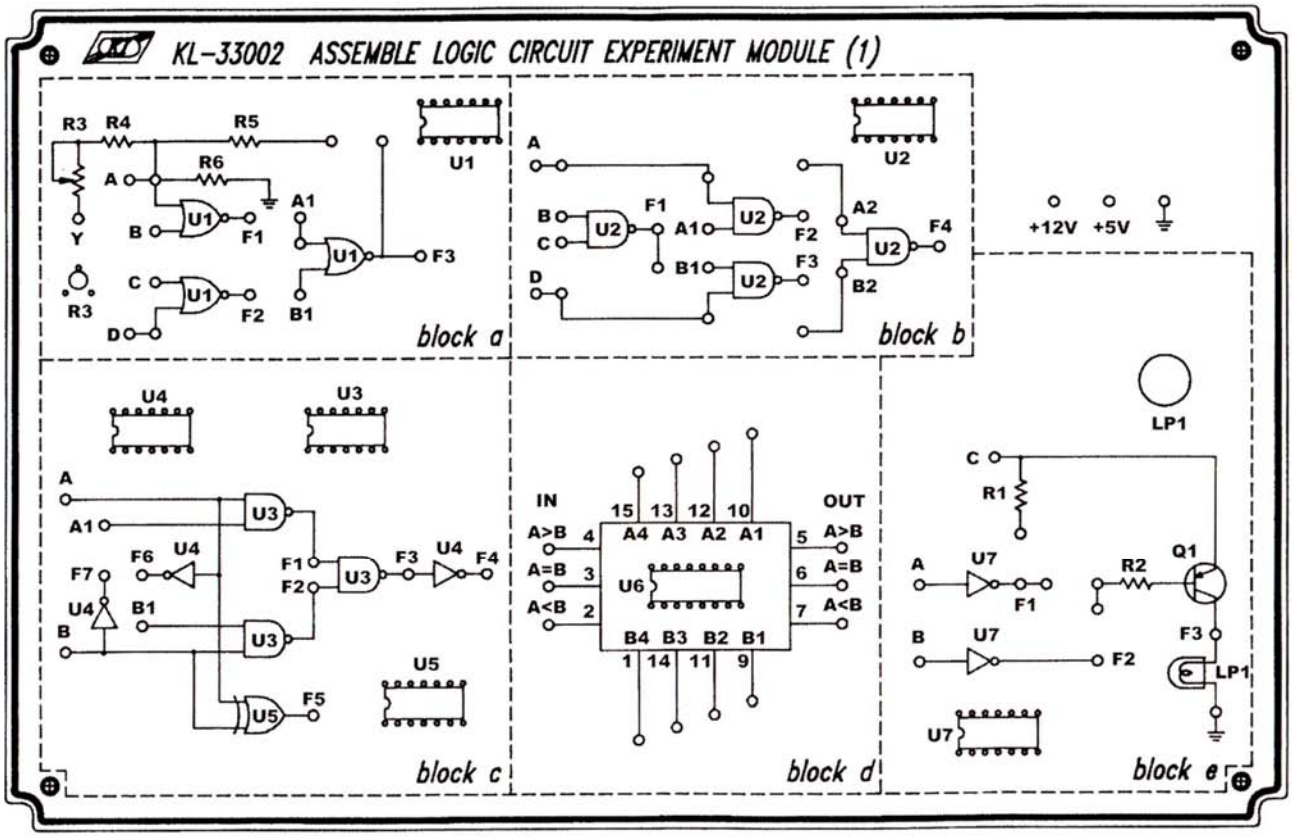
b) Implement the second XOR gate with four **NAND** gates.

3) Design a 5-bit comparator assuming that you have a 4-bit comparator and a 1-bit comparator. The input of the 1-bit comparator should be the MSB of the 5-bit input. Follow this procedure in your design:

a) Find the truth table of the 1-bit comparator shown in page 9. Is it active HIGH or active LOW?

b) Design a circuit that accepts the outputs of the 1-bit and 4-bit comparators as inputs (the circuit will have 6 inputs) and produces 3 active HIGH outputs (A>B, A=B, A<B). Derive the truth table illustrating the behavior of your circuit.

4) Make sure that you are able to implement the circuits you have designed in parts **1-3** above using only the blocks of module KL-33002 shown below. This module contains one 4-bit comparator and some NAND, NOR, … gates.



LAB

***Note**: – When you finish each part of the Lab procedure, before proceeding to the next part, make sure that your TA or instructor checks your results and you get the mark for that part.*

– Record the results you obtain in each part in your lab report.

1) Implement the divide by 3 circuit you've designed in *PRELAB* with gates available on KL-33002 ***blocks a, b, c*** and ***e***. Verify the circuit operation by applying different BCD inputs. Find the outputs for don't-care inputs and write them in your report.

2) Complete the 1-bit comparator circuit on ***block c*** and verify its operation. Wire the circuit of the 5 -bit comparator you've designed in *PRELAB* and observe the outputs.

3) Implement an XOR gate with four NAND gates available on ***block b*** and verify its operation. Implement the full-subtractor you've designed with NOR gates using ***block a*** and observe the outputs. You may use NOT gates available on other blocks and XOR gate available on ***block c***.

Experiment 3

Code Converters and Decoders

Objective: Combinational circuit design, Introduction to binary codes and conversion from one code into another

***Required Equipment*:** Breadboard, IC type 7486, KL-33004 and KL-33005 Modules.

BACKGROUND INFORMATION

As you know, every piece of data in digital world should be expressed as a binary code. When we deal with numbers, the conventional binary system (i.e., the radix 2 number system) is the most common; but there are many applications in which other codes are preferable. One such code is the Gray code. 4-bit binary numbers and their corresponding Gray code are shown in the following table:

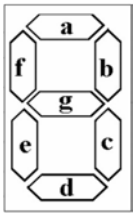
Decimal	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

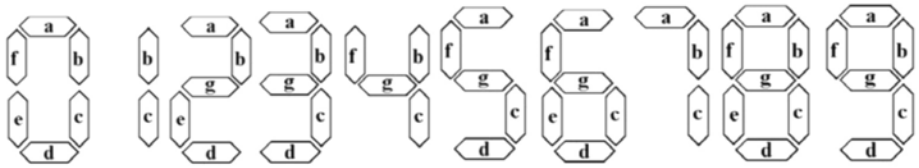
Another common number code is the BCD, in which decimal digits 0 ~ 9 are replaced by their binary equivalent (0000 ~ 1001).

There are, therefore, many occasions in which conversion from one binary code into another is necessary in a digital system; for instance, conversion from binary to Gray code (or vice versa), and decoding a BCD digit into 7-segment (as described below).

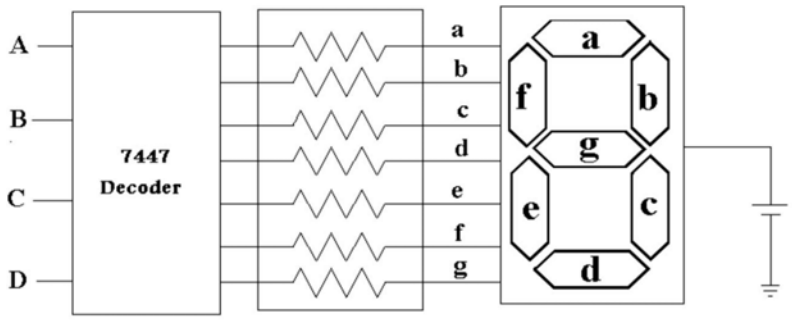
When we want to display a BCD digit with a 7-segment display (a set of seven LEDs as in the following figure), the equivalent seven-segment code (*a*, *b*, *c*, *d*, *e*, *f*, *g*) of the BCD digit should first be

obtained. This code implies which LEDs should be ON for the desired digit (a=0 indicates that LED (a) is OFF and a=1 means it is ON).





A BCD-to-7-segment decoder IC, such as 7447 or 7448, could be used for this purpose. So, BCD digits can be displayed on 7-segment indicators by using a configuration like this:



Limiting resistors must be used at the inputs to prevent high currents into the 7-segment display.

PRELAB

1) Design a combinational circuit that converts a four-bit binary number to its corresponding Gray code (use Karnaugh map for simplification). You should be able to implement it with only XOR gates.

2) Consider a BCD-to-7-segment decoder with inputs D, C, B, A (D is the most significant bit) and outputs *a*, *b*, *c*, *d*, *e*, *f*, *g*. Find the output *(g)* in terms of D, C, B, and A (The unused input combinations should be regarded as don't-care conditions).

3) Design a circuit that generates even parity for a 5-bit binary number (The output is '1' when the number of ones in the input is odd. Otherwise it is '0'). Implement your design using XOR gates.

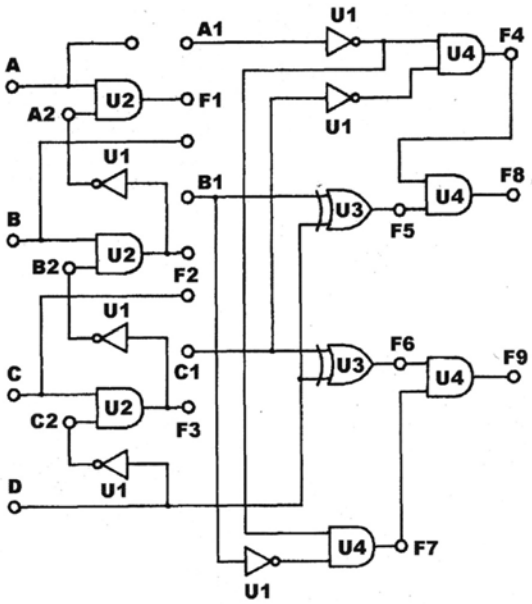
4) Design a combinational circuit that checks if a BCD digit is a nonzero multiple of 3 (that is, the output is '1' if the input is 3, 6 or 9) using a 4x16 (or a 4x10) decoder:

a) Implement your design using one 4x16 decoder with active HIGH outputs and one OR gate.

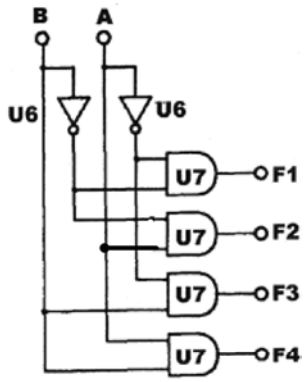
b) Implement your design using one 4x16 decoder with active LOW outputs and one AND gate.

5) a) Design a combinational circuit that generates the 9's complement of a BCD digit. You can use ***LogicWorks*** to verify your design.

b) Implement the circuit only with gates available on ***blocks a*** and ***block c*** of KL-33005 (shown below). Note that **F1~F9** are outputs of the gates and you cannot apply the inputs to them.



block a

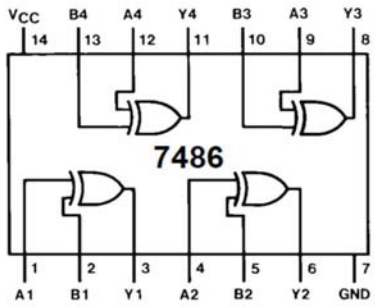


block c

LAB

Note: – *When you finish each part of the Lab procedure, before proceeding to the next part, make sure that your TA or instructor checks your results and you get the mark for that part.*
– *Record the results you obtain in each part in your lab report.*

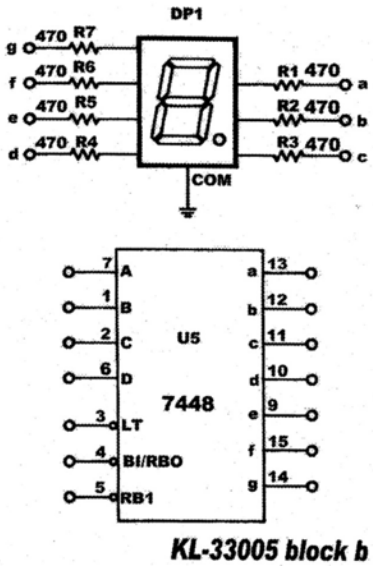
1) Using one 7486 IC implement the binary-to-Gray code converter that you designed in *PRELAB*. Connect the 4 inputs to switches **SW0**-**SW3** and the 4 outputs to **L0**-**L3**.



2) There are a 7448 decoder and a 7-segment display on module KL-33005, ***block b***. Connect switches **SW0**-**SW3** to inputs A, B, C, D of 7448 and its outputs to 7-segment display.

a) With LT, BI/RBO, and RBI open (or HIGH), observe and record the displayed pattern for all inputs from 0000 to 1111.

b) By applying appropriate inputs and observing the display, can you determine the function of LT (Lamp Test), BI/RBO (Blanking Input/Ripple Blanking Output), and RBI (Ripple Blanking Input) pins of 7448?



KL-33005 block b

There are 4 seven-segment displays on the top right corner of the KL-31001 main unit (see part (10) on Figure 2, page 5), with five inputs DP (decimal point), A, B, C, D below each. You can use them to display BCD inputs and/or outputs. The required BCD-to-7-segment decoder is located inside the unit.

3) Implement the 4-input/4-output *9's complementer* combinational circuit designed in *PRELAB*, only with gates available on ***block a*** and ***block c*** of KL-33005. Connect inputs simultaneously to both data switches **SW0**-**SW3** and to inputs A, B, C, and D of the seven-segment display **D0**. Connect outputs to seven-segment **D2**.

4) Implement the parity generator circuit using module KL-33004, ***block a***.

5) Implement the circuit of part **(4)** of *PRELAB* using the 4x10 decoder on ***block c*** and some gates of ***block a*** of module KL-33004.

QUESTIONS

1) Name some applications of Gray code.

2) Check the datasheet to see the application of RB/RBI and RBO pins of 7448. Have you obtained the same result in the Lab?

Experiment 4

Multiplexers and ALU

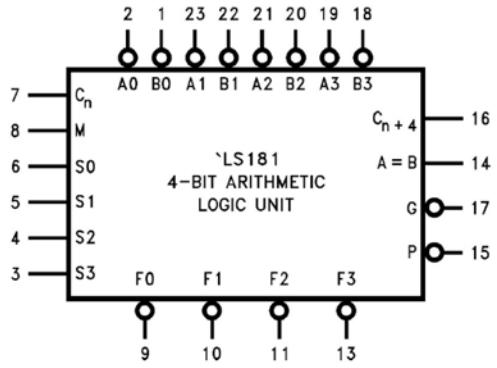
Objective: Combinational circuit design with multiplexers, Introduction to circuits for arithmetic/logic operations

***Required Equipment*:** Breadboard, KL-33003 Module, IC types 74151 and 74157 (or 74257).

BACKGROUND INFORMATION

- Multiplexers have many applications in digital system design. Basically, they are used to select a line (or a bus) of data, amongst several others. They are also utilized to implement different combinational circuits. Refer to your Logic Circuit course textbook for a review of this topic (e.g., section 4.11 of Mano, 4th ed.).

- ALU (Arithmetic and Logic Unit) is a basic part in the central processing unit of all computers. ALU is a combinational circuit that can perform many arithmetic and logic operations. 74181 IC is an example of an ALU, which was used as the arithmetic/logic core in the CPUs of many historically significant minicomputers. This IC has a mode control input (M) that selects between arithmetic/logic modes, and four select inputs ($S_3S_2S_1S_0$) that determine which operation should be performed. The input and output data can be regarded as active-high (positive logic) or active-low (negative logic); that is, low-voltage level ('L') is considered to be logic '0' and high-voltage level ('H') to be logic '1' (active-high), or 'L' to be logic '1' and 'H' to be logic '0' (active-low). Pin-out and a part of the function table of 74181 are shown below:



Pin Names	Description
$\overline{A}0$–$\overline{A}3$	Operand Inputs (Active LOW)
$\overline{B}0$–$\overline{B}3$	Operand Inputs (Active LOW)
S0–S3	Function Select Inputs
M	Mode Control Input
C_n	Carry Input
$\overline{F}0$–$\overline{F}3$	Function Outputs (Active LOW)
A = B	Comparator Output
$\overline{G}$	Carry Generate Output (Active LOW)
$\overline{P}$	Carry Propagate Output (Active LOW)
C_{n+4}	Carry Output

Mode Select Inputs				Active LOW Operands & F_n Outputs		Active HIGH Operands & F_n Outputs	
				Logic	Arithmetic (Note 2)	Logic	Arithmetic (Note 2)
S3	S2	S1	S0	(M = H)	(M = L) (C_n = L)	(M = H)	(M = L) (C_n = H)
L	L	L	L	$\overline{A}$	A minus 1	$\overline{A}$	A
L	L	L	H	$\overline{AB}$	AB minus 1	$\overline{A} + B$	A + B
H	H	L	L	Logic 0	A plus A (Note 1)	Logic 1	A plus A (Note 1)
H	H	L	H	$A\overline{B}$	AB plus A	$A + \overline{B}$	(A + B) plus A
H	H	H	L	AB	$A\overline{B}$ minus A	A + B	$(A + \overline{B})$ plus A
H	H	H	H	A	A	A	A minus 1

PRELAB

1) Design a combinational circuit that evaluates the votes of a committee with four members; that is, it accepts four binary inputs (YES or NO vote of each member) and at the output produces accept ('1') by majority or reject ('0'). For example: 1 0 0 1 → 0, or 1 1 1 0 → 1. Implement your design using only an 8x1 multiplexer.

2) i) Find the structure of a full-adder and draw it.

ii) Show how an *n*-bit ripple-carry adder is constructed using *n* full-adders (Use block diagrams).

iii) If all gates have equal delays of 10 ns, what is the delay of a 16-bit adder (How long does it take for a valid output to appear after applying the inputs)? What is the maximum frequency of operation of this adder?

3) i) The delay of the above adder increases linearly with *n* and cannot be tolerated in high-speed systems. Can you suggest a way to reduce this delay?

ii) What is the delay of the adder that you have designed in the previous part?

4) Complete the following table with binary numbers.

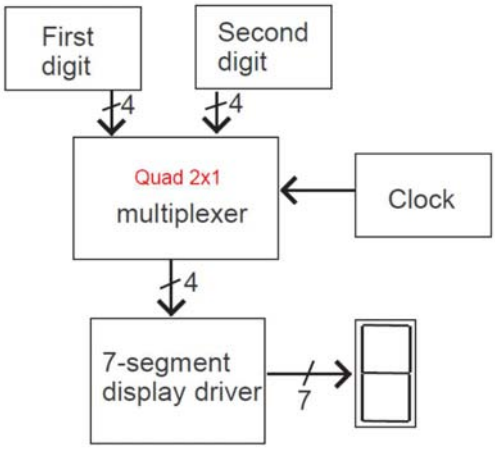
		Active HIGH				Active LOW			
A	**B**	**A−B**	**Borrow**	**A+B**	**Carry**	**A−B**	**Borrow**	**A+B**	**Carry**
L L L L	H H H H								
L H L H	L H H L								
H L H L	H L H L								
H H H H	L L H H								

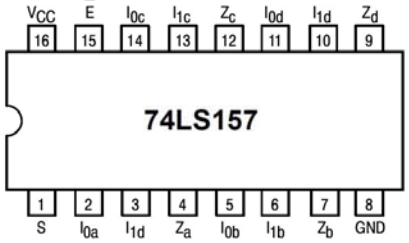
5) Read the *LAB* part below to get an idea of what you should do in your lab session.

LAB

> ***Note***: – *When you finish each part of the Lab procedure, before proceeding to the next part, make sure that your TA or instructor checks your results and you get the mark for that part.*
> – *Record the results you obtain in each part in your lab report.*

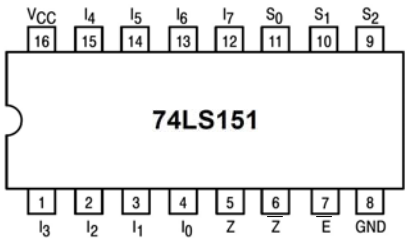
1) i) Design a circuit that displays two decimal digits alternatively on a single 7-segment (e.g., for 0.5 seconds it displays '5' and for the next 0.5 s, it outputs '2'). You can use a circuit like the one shown below. You may use 74157, which contains four 2x1 multiplexers with common select pin.

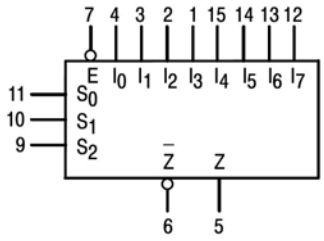




ii) The human eye cannot track very rapid motions (like rotation of a fan), because there is a phenomenon in the eye, called persistence of motion, by which an afterimage persist on the retina for a fraction of a second (In the early days of cinema and film innovation, it was scientifically determined that a frame rate of less than 16 frames per second (frame/s) caused the mind to see flashing images). This means that eye samples the images with a limited frequency and we cannot see changes in the image with a frequency higher than that.
Use the above circuit to estimate the sampling frequency of your eye: Increase the clock frequency slowly to a degree such that you see all the 7 LEDs 'ON' simultaneously (see an 8).

2) Implement the majority voting circuit that you designed in *PRELAB*, using a 74151 IC.





3) In the ALU on KL-33003 module, connect outputs **F_0~F_3** to inputs **A_0~A_3** . Set M=**'L'**, C_n=**'H'**, and $S_3S_2S_1S_0$=**'HHHH'**. Observe C_{n+4} on the oscilloscope and draw its waveform in your Lab report. What is the frequency of this waveform? Can you give an estimate of the addition delay of ALU from this frequency?

In your Lab report, discuss about what you have done above, and derive a relation between the frequency of the waveform at C_{n+4} and the addition delay of ALU. Consider both active-high and active-low cases. Compare your result with the data on the IC datasheet.

Can you specify the addition mechanism (ripple-carry adder or adder with carry look-ahead)?

4) **i)** Set $S_3S_2S_1S_0$=**'LHHL'** and M=**'L'**. Apply the inputs of the *PRELAB* table and find the outputs (Active-LOW and Active-HIGH) with both C_n=**'H'** and C_n=**'L'**. What is the difference in the output between the two cases of C_n=**'H'** and C_n=**'L'**?

ii) Set $S_3S_2S_1S_0$=**'HLLH'** and M=**'L'** and repeat step **(i)**.

iii) In subtract mode (Active-LOW and Active-HIGH), determine when carry-out becomes logic "1". What is the application of carry-out in subtract mode?

Mode Select Inputs				Active LOW Operands & F_n Outputs		Active HIGH Operands & F_n Outputs	
				Logic	Arithmetic (Note 2)	Logic	Arithmetic (Note 2)
S3	S2	S1	S0	(M = H)	(M = L) (C_n = H)	(M = H)	(M = L) (C_n = H)
L	L	L	L	$\overline{A}$	A	$\overline{A}$	A
L	L	L	H	$\overline{AB}$	AB	$\overline{A}+\overline{B}$	A + B
L	H	H	L	$\overline{A}\oplus\overline{B}$	A minus B	$A\oplus B$	A minus B minus 1
H	L	L	H	$A\oplus B$	A plus B plus 1	$\overline{A}\oplus\overline{B}$	A plus B
H	H	L	H	$A\overline{B}$	AB plus A plus 1	$A+\overline{B}$	(A + B) plus A

Experiment 5

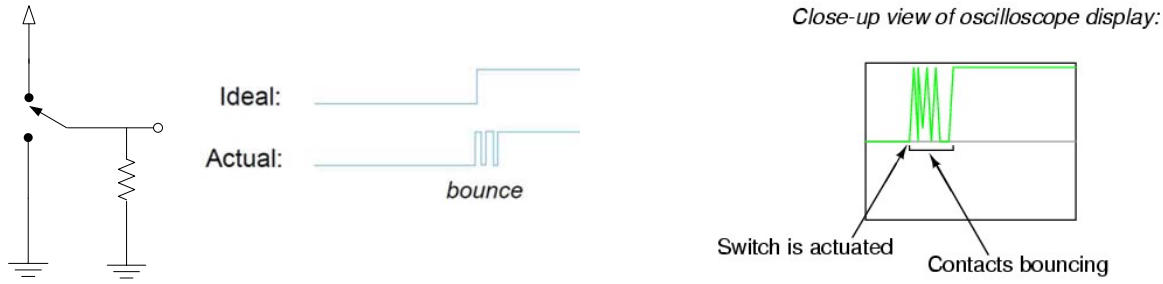
Latches, Flip-Flops, and Registers

Objective: Introduction to structure and operation of latches and flip-flops, (shift) registers and simple storage cells, and switch debouncing.

***Required Equipment*:** KL-33008 Module, Oscilloscope.

BACKGROUND INFORMATION

Switch Bounce: Input binary information in a digital system can be generated manually by means of mechanical switches. One position of the switch provides a voltage equivalent to logic '1', and the other position provides a second voltage equivalent to logic '0'. Mechanical switches are also used to start, stop, or reset the digital system. In testing digital circuits in the laboratory, the input signals will normally come from switches. A common characteristic of a mechanical switch is that when the arm is thrown from one position to the other, the switch contact vibrates or bounces several times before coming to a final rest. In a typical switch, the contact bounce may take several milliseconds to die out, causing the signal to oscillate between 1 and 0 because the switch contact is vibrating.



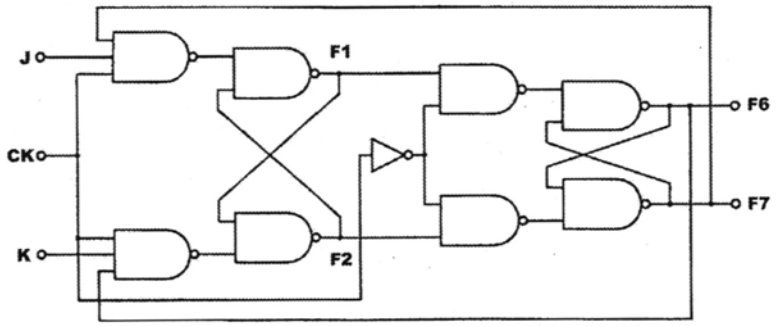
Switch bounce can cause some problems in digital circuits and debouncing is necessary in many cases. In this experiment you will try to design a simple debouncer.

PRELAB

1) **(i)** Draw the structure of an SR-latch with **NAND gates**.
(ii) Show how a D-latch is obtained from SR-latch.

2) **(i)** What is the difference between a latch and a flip-flop (FF)?
(ii) Show how a D-FF is obtained from two D-latches.
(iii) Show how JK-FF and T-FF are obtained from D-FF.

3) Some older JK flip-flops use the following structure. Check its operation to see if it is exactly the same FF as the one you drew above. Can you find the difference (if any)?



4) Check the datasheet of the shift register IC 7495 and describe how it can shift the data 'from right to left'.

LAB

***Note**: – When you finish each part of the Lab procedure, before proceeding to the next part, make sure that your TA or instructor checks your results and you get the mark for that part.*
– Record the results and waveforms you obtain in each part in your lab report.

1) Consider the cross-coupled NAND gate (SR-latch) on ***block d*** of module KL-33008.

(i) Using the S and R inputs (**A3** and **A4**), alternately set and reset the output and verify the operation of the latch.

(ii) Set both inputs to '**1**' and observe the outputs.

(iii) Return both inputs simultaneously to **'0'** and observe the outputs. Repeat (ii) and (iii) several times and observe the outputs. Do the outputs always reach the same state? Obtain the complete function table of the S'R'-latch.

2) Construct the JK-FF of *PRELAB* (3). Use the pulser switch **SWA** (part (8) on Figure 2, page 5) as the clock and connect **J** and **K** inputs to **SW0** and **SW1** and outputs **F1**, **F2**, **F6**, **F7** to **L1~L4**. Follow the input sequence in the table below and observe and record the output states.

INPUT			OUT			
CK	K	J	F1	F2	F6	F7
0	0	0				
1	0	0				
0	0	0				
0	0	1				
1	0	1				
0	0	1				
0	1	0				
1	1	0				
0	1	0				
0	1	1				
1	1	1				
0	1	1				

3) **(i)** Convert the above structure to a T-FF. Apply a 10 KHz clock to this T flip-flop and observe the output on the oscilloscope and record it for both T=**'0'** and T=**'1'**. What is the output frequency?

(ii) Observe the output and the clock signal simultaneously on the two channels of the oscilloscope. What is the relation between them?

(iii) Discuss the results in your lab report.

4) Consider the cascaded four D-FFs in ***block c*** of the module which constitute a 4-bit shift register. Connect **A** (input) to switch **SW0**, **B** (clear) to **SW1**, **CK** (clock) to pulser **SWA**, and outputs **F1~F4** to **L1~L4**.

(i) Load the shift register with the sequence **'1 1 0 1'** (with **SW1**=**'1'**, use **SW0** as data input and **SWA** as clock; send one pulse to **CK** from **SWA** before proceeding to the next bit). Positive edge of the clock triggers this shift register or negative edge?

(ii) ***Block b*** contains a 4-bit shift register with parallel load and a serial input from left (IC type 7495). Connect the shift register on ***block c*** to the one on ***block b*** and fill the 7495 with the 4-bit content from left to right. You should do this such that the data in ***block c*** is not lost after data transfer is complete.

(iii) Repeat the previous part, but now fill the 7495 from right to left (after four clock pulses it should contain **'1 0 1 1'**).

(iv) Now, by setting **SW1**=**'0'**, clear the shift register in ***block c*** and then bring it back to **'1'**. With **SW0**=**'0'**, disconnect the **CK** from **SWA** and connect it to **SW3**. Now, with **SW0**=**'1'** and **SW3** as clock, try to load one bit of **'1'** into the leftmost flip-flop. What happens?

(v) Explain your observation using the fact about switches that you learned in *PRELAB*.

(vi) To have an ideal single clock pulse from **SW0**, a debouncing circuit must be used to remove the ripples (bounces). Use the S'R'-latch on ***block d*** as a debouncer (How?) and repeat part (ii). Describe your observation. Can you use this circuit to debounce more than one clock pulse?

QUESTIONS

1) What limits the frequency of operation of a flip-flop? Suggest a method to measure the maximum clock frequency at which a flip-flop works properly.

2) Using a Flip Flop with asynchronous $\overline{CLR}$, a 4-bit shift Register, and 100Hz squarewave signal, Design a debouncer circuit with multiple pulse support. Assume that the minimum time between two consecutive pulses from **SW0** are 50ms.

a. Explain how the designed circuit can be implemented using LAB modules.

Experiment 6

Finite State Machines

Objective: Design of a general synchronous digital system (FSM).

***Required Equipment*:** Breadboard, Flip-flop ICs, glue logic (AND, OR, NOT,… gates).

PRELAB

1) ***Sequence detector***: Design a circuit that detects the sequence '010' in a string of bits coming through an input line (The output is '1' when the last received three bits make the pattern '010'; otherwise it is '0'). Use Moore model and only **two** D flip-flops.

2) Design the above sequence detector with a Mealy model (and D flip-flops).

3) The 7474 IC contains two D flip-flops. How many ICs are required for the implementation of each of the above circuits on breadboard?

Check the pin assignment of the 7474 from its datasheet and have it with you in the lab.

Note: *7474 has two different pin assignments. In the laboratory you will use DM74LS74AN. Make sure you have downloaded the correct datasheet.*

4) Using JK flip-flops design a sequential circuit that produces the sequence of digits of your student ID number; e.g., for the ID number 9344561:

$9 \rightarrow 3 \rightarrow 4 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 1 \rightarrow 9 \rightarrow \ldots$ ('$\rightarrow$' means one clock cycle)

Give two different designs for your circuit; one using four flip-flops, and the other using three flip-flops.

The 7476 IC contains two JK flip-flops. Check the pin assignment of the 7476 from its datasheet and have it with you in the lab.

LAB

> ***Note**: When you finish each part of the Lab procedure, before proceeding to the next part, make sure that your TA or instructor checks your results and you get the mark for that part.*

1) Implement the Moore FSM of part **(1)** of *PRELAB* on the breadboard. Use pulser **SWA** as the manual clock and toggle switch **SW0** as data input.

2) Implement the Mealy FSM of part **(2)** of *PRELAB* on the breadboard. What is the difference in the behavior of Mealy and Moore models (if any)?

3) (Optional) Wire one of the circuits of part **(4)** of *PRELAB* on the breadboard.

Experiment 7

Stopwatch (Chronometer)

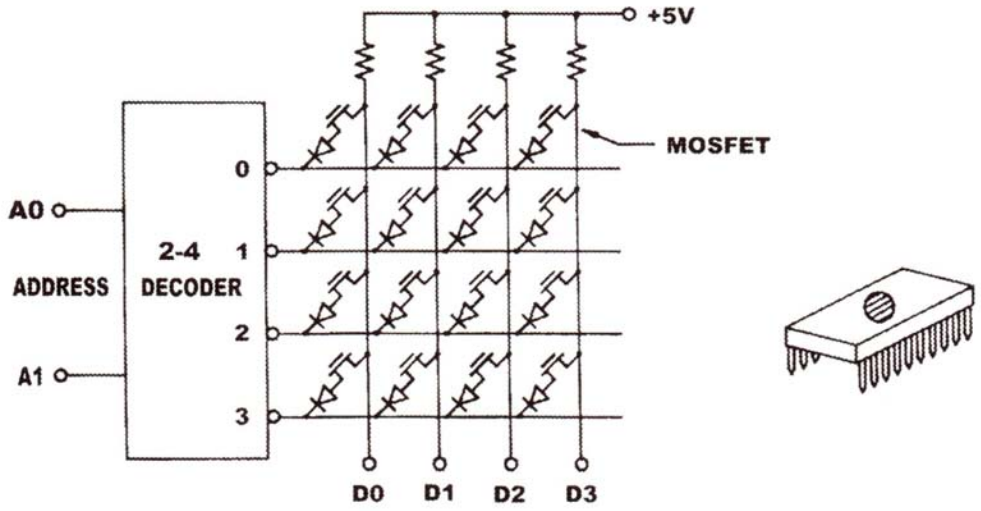
Objective: Introduction to counters and the concept of frequency division; Introduction to EPROM.

***Required Equipment*:** KL-33010 Module.

BACKGROUND INFORMATION

Erasable Programmable ROM (EPROM)

Although PROMs (Programmable Read Only Memory) allow users to write-in data by themselves, it is a one-time-only device and its use is very inconvenient and uneconomical if several modifications are necessary. Erasable PROMs (EPROM) is a kind of non-volatile memory in which the stored data can be erased by exposing it to ultra-violet (UV) light. The figure below shows the general schematic of an EPROM.



The most distinguishable mark of an EPROM is the quartz window on the top surface. On the inside, it is the Metal-Oxide Semiconductor Field Effect Transistor (MOSFET) that sets EPROM apart from other memories. In a brand new EPROM, no bias exists at any of the MOSFETs so the MOSFETs are not conductive; outputs **D0**-**D3** are initially at logic "1". If logic "0" is required at a certain bit, an EPROM writer (programmer) must be used to apply a certain voltage (depend on EPROM specifications, typically 21V or 25V) to this particular bit, injecting electrons into the gate. Since the gates are covered with insulating materials, the electrons are trapped inside. The gates now become negatively charged, creating a path in the MOSFET and making it conductive, or logic "0."

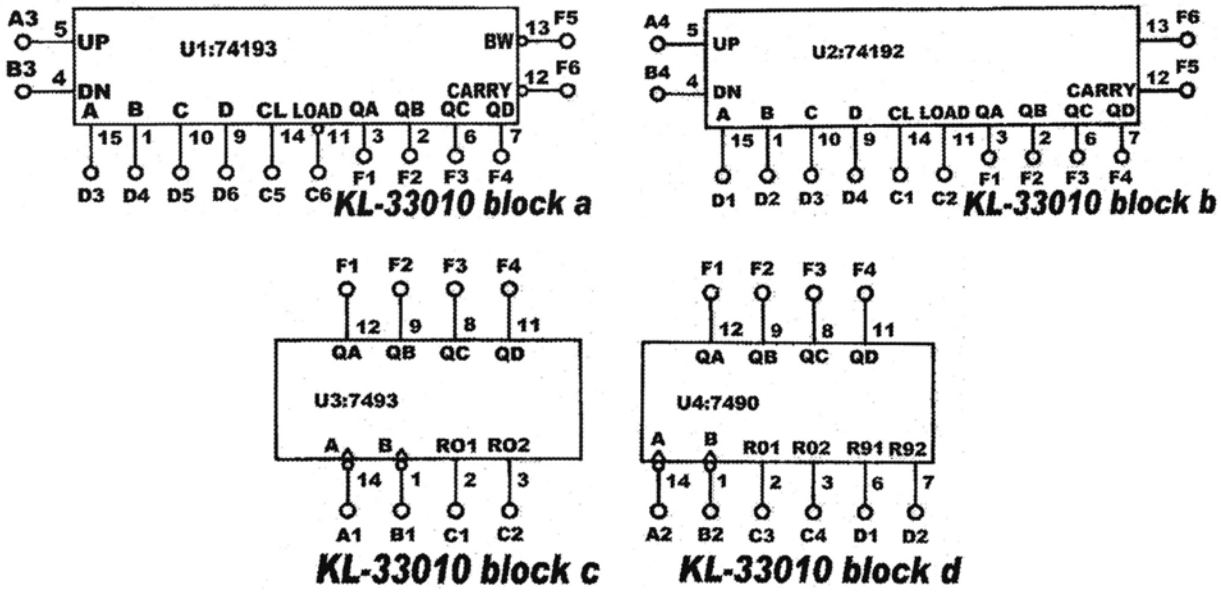
Exposure to ultraviolet light, through the quartz window, will erase data stored in the EPROM. The duration of exposure depends on the length of data stored, typically anywhere from several minutes to 30 minutes. When the MOSFETs are exposed to ultraviolet light, the trapped electrons gain enough energy to escape the insulating material, thus making the MOSFET not conductive.

In this experiment you will become familiar with applications of counters in constructing a stopwatch and in addressing an EPROM.

PRELAB

1) Explain how a counter can be used as a frequency divider.

2) ***Blocks a, b, c***, and ***d*** of KL-33010 contain a synchronous 4-bit binary counter, a synchronous decade counter, and asynchronous (ripple) binary/decade counters, respectively (figures below). For a detailed description of the operation of these counters refer to datasheets.



Using these counters, design a circuit that accepts an 80 Hz clock and outputs the sequence 00~99 with a rate of 10 Hz (i.e., counts from 0.0 to 9.9 seconds in steps of 0.1 s).

Hint: You can first use a divide-by-8 counter to obtain a 10 Hz clock, and then apply it to two cascaded decade counters.

3) Using *LogicWorks*, simulate your design to make sure it works correctly. Print your design and hand in with your pre-lab report.

4) ***Block e*** of KL-33010 contains an EPROM (IC 2764) which has 13 address lines and 8 data lines. Find the total capacity of this EPROM in Kb.

LAB

***Note**: When you finish each part of the Lab procedure, before proceeding to the next part, make sure that your TA or instructor checks your results and you get the mark for that part.*

1) Implement the stopwatch that you have designed in *PRELAB*.

2) **(i)** In the ***block e*** of KL-33010 connect, **S1~S3** to data switches **SW1~SW3**; outputs **D7~D0** to logic indicators **L8~L1**, respectively.

(ii) The state of input **S1S2** will determine the outputs. Observe the output for different values of **S1S2** listed below. How does the output change when **S3='0'** and when **S3='1'**?

1: **S1S2='00'** (BILLBOARD 1)
2: **S1S2='01'** (BILLBOARD 2)
3: **S1S2='10'** (TRAFFIC LIGHT)
4: **S1S2='11'** (FLASHING YELLOW)

(iii) Complete the table below of EPROM contents for TRAFFIC LIGHT mode. Note that each output pattern that you see occurs in two consecutive addresses. Explain how counters change the address in each mode.

A3	A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	1	0	0	0	0	1	0	0
0	0	0	1	1	0	0	0	0	1	0	0
0	0	1	0								
0	0	1	1								
0	1	0	0								
0	1	0	1								
0	1	1	0								
0	1	1	1								
1	0	0	0								
1	0	0	1								
1	0	1	0								
1	0	1	1								
1	1	0	0								
1	1	0	1								
1	1	1	0	0	0	1	0	0	0	1	0
1	1	1	1	0	0	1	0	0	0	1	0

QUESTION

1) Can we leave an EPROM out in the Sun to erase it? How long will it take?

2) **(Optional)** Modify the circuit you designed in *PRELAB* such that it includes a single pulser that starts and stops the counting process; that is, when the pulser is pressed the counting starts and when it is pressed again counting is stopped. Two seven-segment displays show this time duration until another pulser (as a reset) is pressed, which clears the stopwatch to 00. Simulate your design with *LogicWorks*.

Experiment 8

EEPROM

Objective: Understanding the theory, structure and applications of EEPROMs.

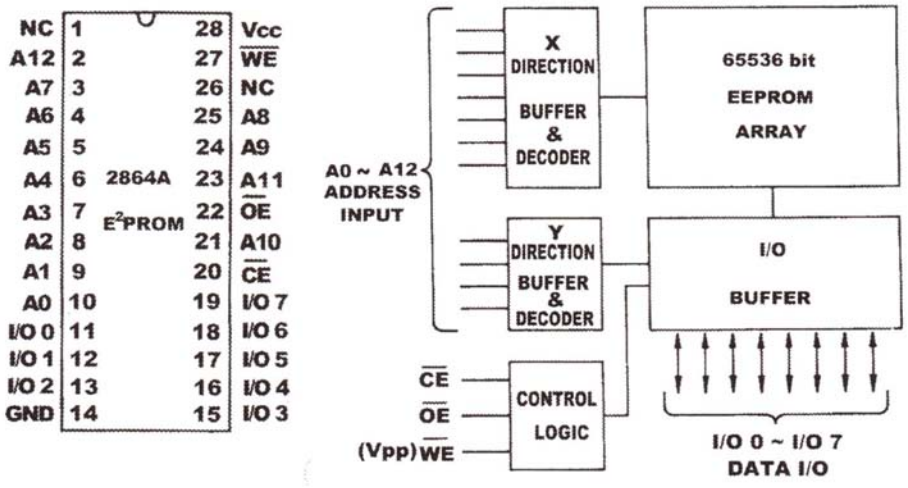
***Required Equipment*:** KL-33011 Module, Breadboard, 4-bit counter IC (7493 or 74193, …).

BACKGROUND INFORMATION

Electrically Erasable PROM (EEPROM)

The only difference between EEPROM (sometimes called E^2PROM) and EPORM is in how they are erased. While EPROM is erased by exposure to ultraviolet (UV) light, EEPROM is erased by applying a voltage; so, it has no quartz window. This characteristic enables EEPROMs to be erased address by address, whereas all data in EPROM will be erased when exposed to ultraviolet light. Therefore, EEPROM is more convenient to use than EPROM. However, EEPROMs have a limited number of write/erase cycles and are usually used for storing tables or charts that are not accessed often and does not change too frequently

The 2864 EEPROM is used in this experiment; its pin assignment and schematics are shown in the following figure.

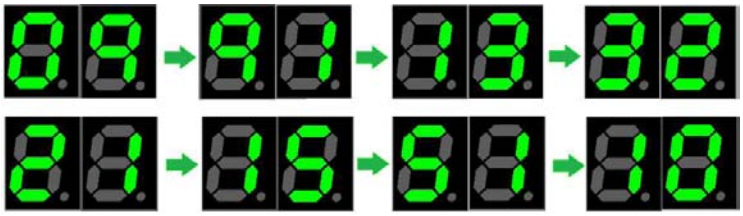


Control pins for 2864 include **$\overline{\text{CE}}$** , **$\overline{\text{OE}}$** and **$\overline{\text{WE}}$** . **$\overline{\text{CE}}$** is *'Chip Enable'*. In order to trigger the IC, **$\overline{\text{CE}}$** must be **'0'**. **$\overline{\text{OE}}$** is *'Output Enable'*; for READ operation, **$\overline{\text{OE}}$** and **$\overline{\text{CE}}$** must both be **'0'** and **$\overline{\text{WE}}$** (*'Write Enable'*) must be **'1'**. For WRITE operation **$\overline{\text{CE}}$** must be **'0'**, **$\overline{\text{OE}}$** must be **'1'**, and **$\overline{\text{WE}}$** must be **'0'**.

PRELAB

1) Design an arbitrary sequence of patterns for the LEDs on the main unit (similar to the TRAFFIC LIGHT in experiment 7). Write the table of the EEPROM contents. The table can have 16 different rows and 8 columns.

2) Design a scrolling text board with two seven-segment displays and EEPROM. This scrolling text board should display your own student number. For example, if your student number is 9132151 the output should display the following sequence:



Your table should have 8 columns and 8 rows. In each row two BCD numbers (in the above example, 0000 1001 in the first row) should be stored. Write the table of the EEPROM contents that generates the sequence of your own student number.

3) Explain how the EEPROM can go automatically through 16 different addresses.

LAB

Follow the procedure described below to program the EEPROM and observe the result:

1) Connect address lines **A0**-**A7** to DIP switches 2.0-2.7; connect data lines **D0**-**D7** to DIP switches 1.0 -1.7, $\overline{\textbf{CE}}$ to data switch **SW0**, $\overline{\textbf{OE}}$ to **SW1**; $\overline{\textbf{WE}}$ to **SW2**, and **A8**-**A10** to '**0**'. If $\overline{\textbf{CE}}$ = **'1'** the 2864 is disabled.

2) Write the pattern you have designed in *PRELAB* part **(1)** in the memory addresses (**A0**-**A7**) from 0000 0000 to 0000 1111. To write in an address:

a. set $\overline{\textbf{OE}}$ = **'1'**;
b. set the address and data using DIP switches;
c. set $\overline{\textbf{WE}}$ = **'0'**, then set $\overline{\textbf{WE}}$ = **'1'**;
d. Repeat to write data in the next address.

3) To check what you have stored in the EEPROM, connect **F0**-**F7** to **L0**-**L7**; set $\overline{\textbf{OE}}$ = **'0'** and $\overline{\textbf{WE}}$ = **'1'** and change the address, the stored data will be displayed on the LEDs.

4) Write the scrolling student number table that you have designed in the *PRELAB* in a different address (for example, from 0001 0000 to 0001 1111; only change the four LSBs). Write each row of the table in two consecutive addresses to fill 16 rows of EEPROM.

5) Connect **F0**-**F3** and **F4**-**F7** to two seven-segment displays and check the stored data. You should see the pattern described in *PRELAB* on the seven-segment displays as you change the address.

6) Remove the 2864 IC from the module KL-330011 using an IC remover.

7) On a breadboard connect the 4 LSBs of the address lines of 2864 to a 4-bit counter (e.g., 74193). Connect the data lines **D0**–**D7** to LED indicators and see the pattern that is displayed on the LEDs.

8) Connect the data lines **D0**–**D7** to two seven-segment displays. By applying appropriate input to address lines of 2864 see your student number scrolling on the seven-segment displays.

Experiment 9

Digital Transmission of Analog Signals

Objective: Introduction to serial and parallel data transmission; Introduction to applications and interfacing of shift registers; Understanding basic concepts and theories of D/A and A/D conversion.

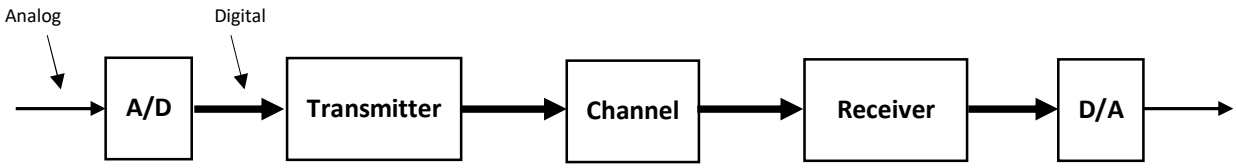
***Required Equipment*:** Breadboard, IC types DAC0800, ADC0804, 74194 (×3), 74193.

BACKGROUND INFORMATION

Many of information bearing signals, such as speech, video, etc., are analog in nature; that is, they are functions of the continuous variable t and for any $t = t_1$ their value can lie anywhere in some interval, say $-A$ to A. However, it is possible to transmit the analog information in a digital format .

The main advantage of using digital signals over analog signals is that the precise signal level of the digital signal is not vital. This means that digital signals are fairly immune to the imperfections of real electronic systems (e.g., *noise* and nonlinearities), which tend to spoil analog signals .

To transmit an analog signal in a digital form it is necessary to convert it into a digital signal at the transmitter using an Analog-to-Digital Converter (ADC or A/D), and then convert the received digital signal to an analog signal at the receiver using a Digital-to-Analog Converter (DAC or D/A). The block diagram of such a system is shown below.



Analog-to-Digital Conversion

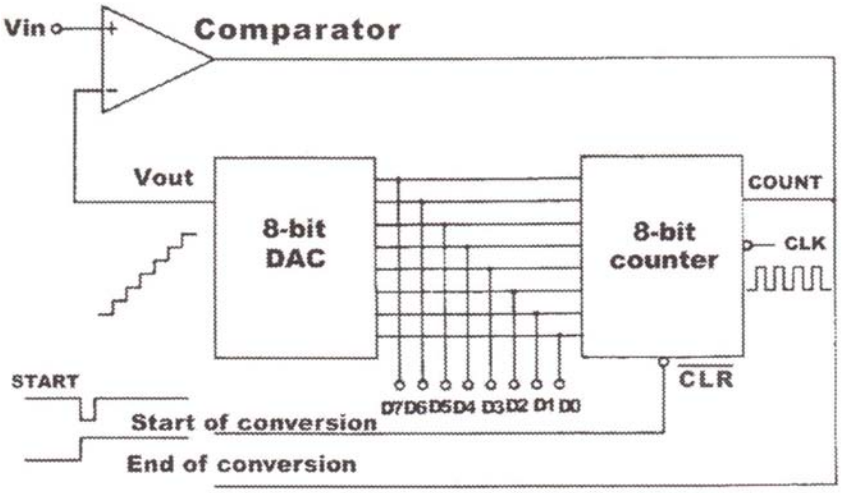
A typical ADC has a single analog input and a parallel, multibit binary output. The number of output bits will determine the ADC's resolution. ADCs are available commercially from 4 to 20 output bits. ADCs with higher number of output bits will have higher resolution .

A number of different circuits have been devised to perform A/D conversion. Four most common methods are

1. Counter-Ramp Feedback ADC

2. Successive Approximations Converter
3. Dual Slope ADC
4. Flash Comparator ADC

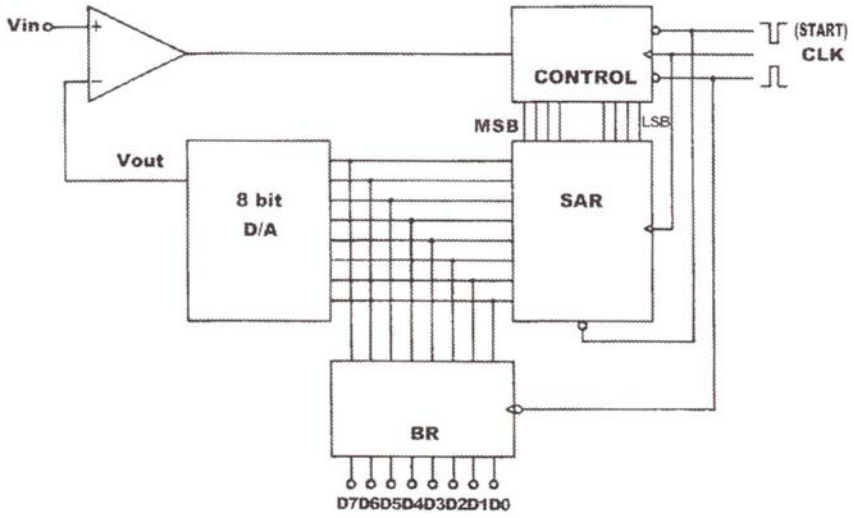
Counter-Ramp Feedback ADC is the simplest but least used type of ADC. Its block diagram is shown in the figure and includes a DAC, a counter and a comparator. The analog input signal (Vin) to be converted is applied to one input of the comparator. The other input to the comparator is the output of the DAC (Vout). The DAC is driven by the counter and incremented by a clock signal.



Counter-ramp feedback ADC

When an analog signal is applied to the input, a start pulse is applied to the counter. This start pulse resets the 8-bit binary counter so that it begins counting from "0". At this time, output of the 8-bit DAC (Vout) is also "0" so the comparator's output is "1" and the clock signal will enable and increment the counter. As the counter is incremented, the DAC will generate a stair-step output voltage (Vout). As soon as Vout reaches Vin, the comparator output will switch to "0", stops the counter and the conversion. Output of the counter at this point is the digital equivalent of the analog input. Negative edge of the comparator output is used to signal the end of conversion.

The **Successive Approximations Converter** is an improved, faster version of the counter-ramp feedback counter. Block diagram of a typical successive approximations converter is shown below. Just like the counter-ramp feedback converter circuit, the circuit contains a DAC and a comparator, but the counter driving the DAC is replaced by a special sequential circuit called "Successive Approximations Register" or "SAR."



Successive Approximations ADC

When a conversion is initiated the SAR is reset; but immediately the SAR's MSB is set to **'1'** and all the other lower bits are set to **'0'**. The MSB causes the DAC output to be one-half of the input reference voltage. The comparator will then compare the DAC output (**Vout**) with the analog input (**Vin**). If **Vout** is larger than **Vin**, the MSB is turned off to **'0'** and the next MSB is turned on (**'1'**). If **Vin** is larger than the DAC output, the MSB will remain set and the next MSB is set to **'1'**. Again the DAC output is compared with **Vin** and it will remain set or be reset. The bits in SAR, from MSB to LSB, are continuously set and reset after each comparison until the final binary output is found. When the final binary output has been determined an "End of Conversion" signal is generated by the control circuit and the final output is send to the Buffer Register .

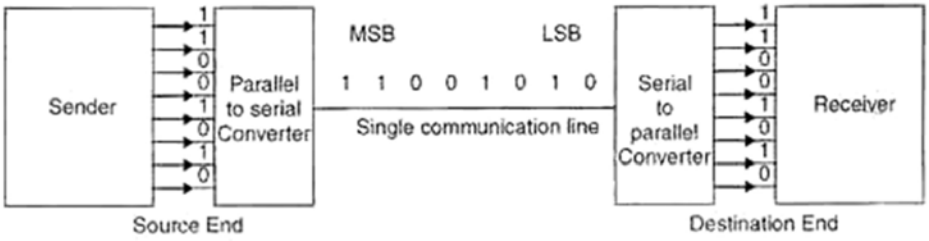
The most significant advantage of the successive approximations converter is its high operating speed, which is the number of bits times the clock period. The successive approximations converter is probably the most popular form of ADC, due to its high operating speed.

Serial and Parallel Data Transmission

Digital data can be transmitted either in a parallel or serial way. The basic difference between a parallel and a serial communication channel is the number of electrical conductors used at the physical layer to convey bits. In parallel communications more than one conductor exists. For example, an 8-bit parallel channel (e.g., a cable with 8 wires) will convey 8 bits (or a byte) simultaneously, whereas a serial channel would convey those same bits sequentially (one bit at a time).

One big advantage of having fewer wires/pins in a serial transmission is the significant reduction in the size, weight, and the complexity of the connectors and the associated costs. Designers of devices such as smartphones benefit from the development of connectors/ports that are small, durable, and still provide adequate performance.

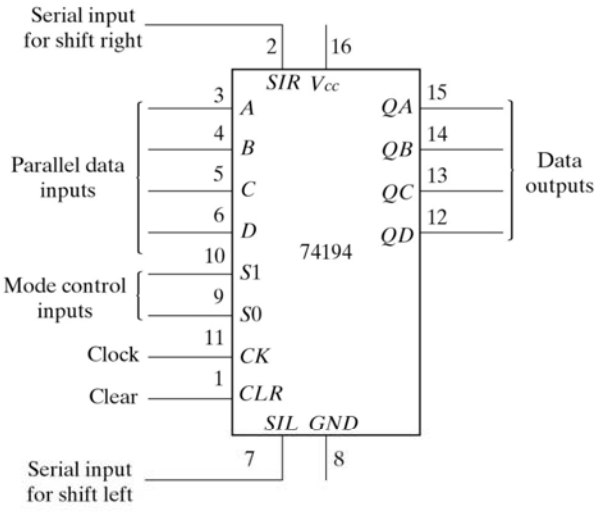
Since the output of ADC is inherently parallel, to transmit the acquired data in a serial way it is necessary to first convert the parallel data into serial. The received data is then converted back into parallel before D/A at the receiver side. Parallel to serial converters (P/S) and serial to parallel converters (S/P) are used for this purpose (see the figure below).



Serial transmission

In this experiment a shift register with parallel input serial output is used in the transmitter to transmit parallel input data in a serial way. At the receiver a shift register with serial input and parallel output is utilized.

IC type 74191 is a 4-bit bidirectional register with parallel load. You can find its internal logic in your textbook (chapter 6 of Mano). The pin assignment to the inputs and outputs is shown in the following figure. The two mode-control inputs determine the type of operation as specified in the function table.

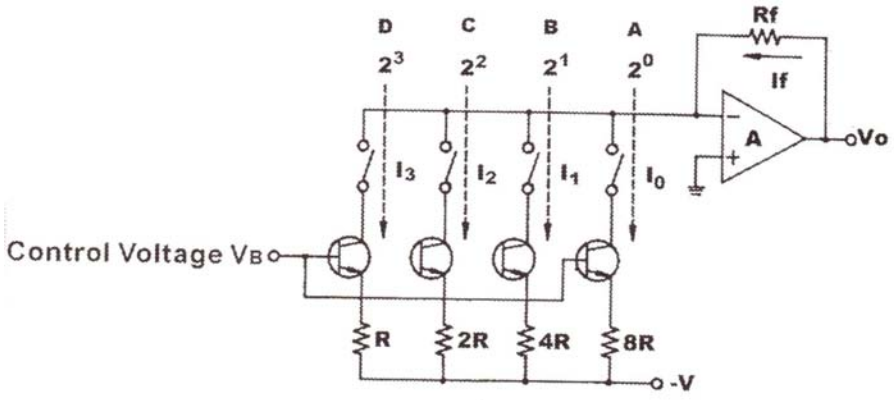


Function table

Mode				
Clear	Clock	S1	S0	Function
0	X	X	X	Clear outputs to 0
1	↑	0	0	No change in output
1	↑	0	1	Shift right in the direction from QA to QD. SIR to QA
1	↑	1	0	Shift left in the direction from QD to QA. SIL to QD
1	↑	1	1	Parallel-load input data

PRELAB

1) The following figure demonstrates a current output type DAC. Explain how it works.



Current output type DAC

2) Assume that a sine wave is applied as the input of a 4-bit ADC and the output of this ADC is connected to the input of a DAC. Sketch the output (the signal recovered from DAC).

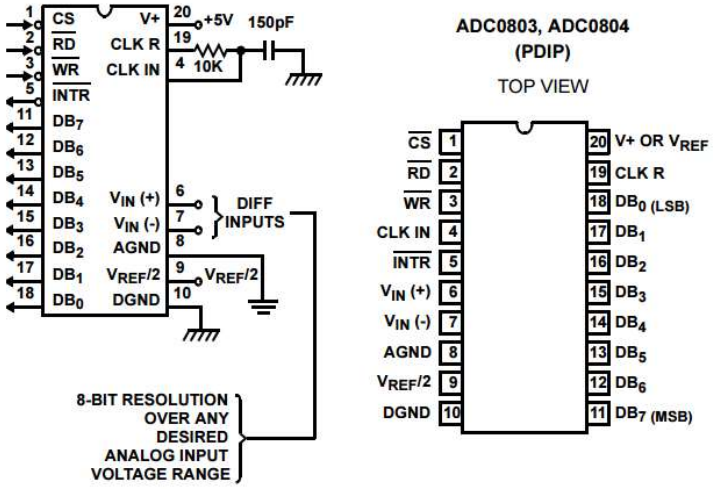
3) Using block diagrams, draw the complete schematics of a system used for serial transmission of analog data.

4) Study the *LAB* part carefully to obtain a good understanding of what you should do in your lab session.

LAB

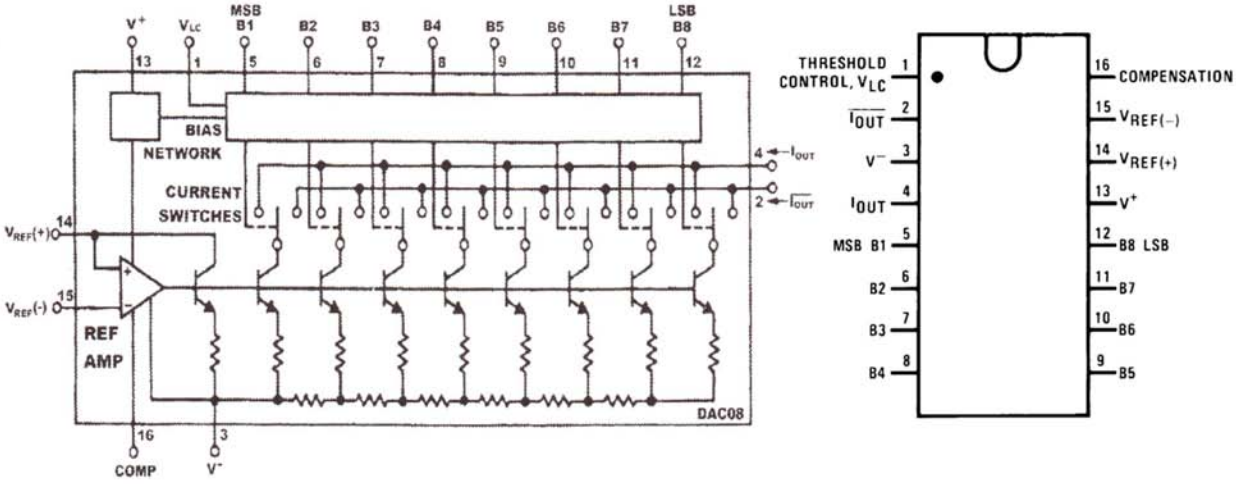
Part I: Parallel data transmission

1) On the breadboard, wire the ADC0804 IC with the following configuration (instead of using an RC circuit to enable self-clocking, you can apply an external clock at about 640 kHz to **CLK IN**). Connect $\overline{\textbf{INTR}}$ to $\overline{\textbf{WR}}$, $\overline{\textbf{CS}}$ and $\overline{\textbf{RD}}$ to GND**,** $\textbf{V}_{\textbf{IN}}\textbf{(-)}$ to GND.



2) Set the function generator to see a 1 kHz sine wave between 0v and 5v (2.5v offset) on the oscilloscope.

3) Insert the DAC0800 IC on the breadboard. Connect pins 1, 2, 15 and **B5**-**B8** to GND, $\mathbf{V^-}$ (pin 3) to −12v, $\mathbf{V^+}$ and $\mathbf{V_{REF}(+)}$ (pin 13 and 14) to +5v .



4) Connect pin 4 to GND with a 470 Ω resistor. The output of this IC is current and the resistor is added to convert the current into voltage (It is also possible to use an operational amplifier. See the datasheet for more information).

5) Connect $\mathbf{V_{IN}(+)}$ of the ADC to the function generator.

6) To perform a parallel digital transmission, connect the 4 MSBs of the ADC ($\mathbf{DB_4}$ – $\mathbf{DB_7}$) to 4 MSBs of the DAC (**B1**-**B4**) and observe the output (pin 4) on the oscilloscope. What will happen if you remove the connection of **DB4**?

Part II: Serial data transmission

Don't remove the previous ICs from the breadboard.

1) Pick two 74194 shift registers. Connect the parallel input of the first 74194 shift register to DIP switches, **S0** to **'1'**, and **S1** to **SW0**. Connect the clear pin of all ICs to **SW1**. Connect the shift right input (**SIR**) of this register to GND.

2) Connect the serial shift right input (**SIR**) of the second 74194 to **QD** of the first IC. Set the second shift register to *shift right* mode.

3) Connect the output of these two shift registers to LED indicators (**L15~L12** and **L9~L6**).

4) Connect the clock of two registers to 1 Hz. Load the first register (using **SW0**) with a 4-bit binary number. Then change its mode to shift right. Observe the transmission of data into the second register.

5) Pick another 74194 and connect all four outputs of the second shift register to parallel inputs of this register. Connect the **S1** and **S0** of this register to **SW0**. Connect outputs of this register to LED indicators (**L3~L0**).

6) Continuously load the first register after the transmission of each 4-bit with new data. Observe the output of the third register .

7) Pick a 74193 counter IC. Put it in count-down mode. Connect its parallel inputs to **'1100'** and its **LOAD** pin to MSB of the output (**A**). Connect the 1 Hz clock simultaneously to three 74194s and 74193. Connect the **S1** of the first register and **S1**, **S0** of the third register to the second bit of the 74193's output (**B**). The transmission of data should be done automatically.

In your lab report clearly explain how serial transmission is performed automatically using the above configuration.

8) Connect the input of the first 74194 to 4-bit MSBs of the ADC and the output of the third shift register to 4-bit MSBs of the DAC. By slowly varying the clock frequency of shift registers from 100 kHz to 1 MHz observe the analog output on the oscilloscope.